Design Patterns SS 2014
Hausaufgabe 5

UNIKASSEL VERSITÄT

Die Aufgaben müssen von jedem Teilnehmer einzeln bearbeitet und abgegeben werden. Die Abgabe muss **bis spätestens Montag 26.05.2014 um 23:59 Uhr** über unser Hausaufgabenverwaltungssystem **https://se.cs.uni-kassel.de/hms/** erfolgen. Die Abgabe ist nur als einzelne *.zip oder *.jar-Datei möglich. Daher müssen alle für eine Abgabe relevanten Daten zu **einem** solchen Archiv kombiniert werden.

Hinweise zur Abgabe:

- Die Hausaufgabe ist in Form von **vier** exportierten Eclipse Projekten abzugeben. Mit Hilfe der Eclipse Export Funktion ist es möglich mehrere Projekte in eine zip-Datei zu exportieren. Ist ein Projekt nicht korrekt exportiert, kann es bei der Korrektur nicht berücksichtigt werden (es bietet sich also an, den Import des exportierten Projektes auszuprobieren).

  WICHTIG: Benennen Sie ihre Projekte für diese Abgabe nach folgendem Schema:

  DPSS14_HA<a>_<b>_<Matrikelnummer>,

  wobei <a> für die aktuelle Hausaufgabe und <b> für die Aufgabennummer steht.
  Beispiel für Aufgabe 1:
  DPSS14_HA5_1_12345678.

Allgemeines
Orientieren Sie sich für die Lösung der Aufgaben an den zugehörigen Übungen und Vorlesungen:
http://seblog.cs.uni-kassel.de/category/currentterm/design-patterns2014/

Die Benotung ergibt sich aus den Hausaufgaben, wobei eine Abgabe ausgelassen werden darf. Für die Note wird die nicht abgegebene Abgabe, beziehungsweise die Abgabe mit der geringsten Prozentzahl, nicht beachtet. Die Endnote ergibt sich aus dem Mittelwert der erreichten Prozentpunkte der übrigen Abgaben.

Die Veranstaltung gilt als nicht bestanden, wenn mehr als **eine Hausaufgabe nicht abgegeben** wurde oder der Mittelwert der zur Benotung herangezogenen Abgaben **weniger als 50%** beträgt.

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Aufgabe 1 Template Pattern

Implementieren Sie ein Datenmodell für die Pfandberechnung der Lagerverwaltung eines Getränkelieferanten unter Verwendung des Template Pattern

- Bilden Sie dabei folgende Teile des Systems ab:
  - GetraenkeKiste
  - Flasche
- Integrieren Sie ein Attribut vom Typ **`int`** in die Klasse GetraenkeKiste:
  - size (es gibt drei Belegungen mit unterschiedlichen Pauschal-Werten für Kisten ohne Flaschen: 6 entspricht 2€, 12 entspricht 3€, 24 entspricht 4€)
- Integrieren Sie zwei Attribute vom Typ **`boolean`** in die Klasse Flasche:
  - leer, kaputt
- Implementieren Sie die Methode *`pfandBerechnen():int`* in die Klasse GetraenkeKiste, welche die Template-Methoden *`getSize():int`* und *`berechneFlaschen():int`* verwendet, um den Pfandwert zu ermitteln.
- Implementieren Sie folgende Template Klassen:
  - WasserKiste
  - ColaKiste
  - Bierkiste
- Schreiben Sie einen Junit-Test in welchem
  - eine sinnvolle Objektstruktur erzeugt wird
  - die Pfandberechnung aller prägnanten Möglichkeiten von Kisten ausgeführt wird
  - sinnvolle Assertions verwendet werden

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Aufgabe 2 Decorator Pattern

Implementieren Sie ein Tresor-System unter Verwendung des Decorator Pattern.

- Bilden Sie dabei folgende Teile des Systems ab:
  - Safe
  - AuthorizationDevice
- Implementieren Sie die Decorator Klassen:
  - NumberPad
  - FingerprintScanner
  - RetinalScanner
  - VoiceRecognition

  Wobei NumberPad der Standard ist. Das bedeutet, dass jeder Tresor mindestens ein NumberPad hat und zusätzlich diverse andere Autorisierungsverfahren zugeschaltet werden. Die Verfahren müssen lediglich schlüssige Systemausgaben tätigen.
- Schreiben Sie einen Junit-Test in welchem
  - eine sinnvolle Objektstruktur erzeugt wird
  - alle prägnanten Kaskadierungen von Authorisierungsverfahren eingesetzt werden
  - sinnvolle Assertions verwendet werden

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Aufgabe 3 Proxy Pattern

Implementieren Sie einen Ausschnitt des Spiels Minesweeper unter Verwendung des Proxy Pattern.

- Bilden Sie dabei folgende Teile ab:
  - MinesweeperGame
  - Map
  - Field
  - Mine
- Implementieren Sie die Klasse Map als Proxy:
  - sie muss eine Methode getFields() anbieten, welche die mit der Map verknüpften Felder zurück gibt.
  - sie muss eine Methode getMines() anbieten, welche die mit der Map verknüpften Minen zurück gibt.
- Implementieren Sie die Klasse Map ein Attribut vom Typ **`boolean`**:
  - init

    Es gibt an ob Felder und Minen vorhanden sind oder ob sie jeweils dynamisch aus einer Textdatei (Fields.sql, Mines.sql) geladen werden.
- Verwenden Sie zum Laden der Felder, beziehungsweise Minen, ein JSON-fähiges Framework ihrer Wahl, beispielsweise SDM-Lib, Gson oder ähnliche.
- Schreiben Sie einen Junit-Test in welchem
  - eine sinnvolle Objektstruktur erzeugt wird
  - die vier Kombinationen (Aufruf von get...() mit init=true und init=false) getestet werden
  - sinnvolle Assertions verwendet werden

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Aufgabe 4 Adapter Pattern

Setzen Sie eine eigene Idee unter Verwendung des Adapter Pattern um.

- Es dürfen keine Beispiele aus dieser Veranstaltung oder von anderen Quellen, wie z.B. Wikipedia, verwendet werden.
- Das System muss anhand eines sinnvollen Tests verifiziert werden.

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de